Laboratorio Nro. 4:
Algoritmos voraces o codiciosos (Greedy algorithms)

Manuela Valencia Toro
Universidad Eafit
Medellín, Colombia
mvalenciat@eafit.edu.co

Laura Sánchez Córdoba
Universidad Eafit
Medellín, Colombia
lsancheszc@eafit.edu.co

Felipe Olaya Ospina
Universidad Eafit
Medellín, Colombia
folayao@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

1. Las estructuras de datos usadas fueron ArrayList y arrays. En el arraylist fueron almacenados los sucesores de cada vértice, y en el array se almacenaron los vértices visitados. Usamos estas estructuras por su fácil accesibilidad a cada posición e igualmente el fácil almacenamiento de información. Este problema lo resolvimos por medio de un algoritmo voraz, en el que por cada vértice se hallan sus sucesores, y de estos el que tenga menos peso, será el próximo nodo a ser visitado y analizado. Esto se hace hasta alcanzar el tamaño del grafo, después de que esto suceda, desde el último nodo visitado se devolverá al inicial, y se retornarán todos los pesos del camino recorrido.
2. Para resolver el problema del agente viajero usando algoritmo voraz, el grafo debe ser conexo ya que todos los nodos deben tener conexión con todos, pues no es posible devolverse si al llegar a un nodo no hay más caminos, y esto generaría un error pues aparte de que el arreglo de sucesores sería nulo,no habría manera de volver al nodo inicial.
3. Las estructuras de datos usadas son los arreglos y ArrayList dado que para este ejercicio se necesita un manejo más sencillo de los datos y su almacenamiento (pues agregar y hallar datos en estos poseen una complejidad O(1)). El algoritmo tiene un solo método el cual en primer lugar, leerá con un scanner la primera línea de cada entrada correspondiente a las rutas, el tiempo y el costo por hora extra.

   Posteriormente, entra a un while que verificará que dicha entrada no sea 0 para las 3 variables (**caso en el cual se acabaría el programa)** y se pasa a una condición que verifica que los valores dados estén en los rangos especificados, en segundo lugar se hace una lectura de las siguientes líneas con un BufferedReader, las cuales hacen referencia a las horas que toman las rutas, dichos valores son convertidos en enteros y sumados a una variable que almacena el tiempo total gastado por las rutas (una para la mañana y otra para la tarde).

   Finalmente, para hallar el costo de las horas extra, se toma el tiempo total de cada ruta, se le restan las horas estipuladas y a lo restante se le multiplica el costo por hora extra, se suman los resultados de la mañana y la tarde; y se almacenan en un ArrayList, cuando el programa acaba, se imprimen los resultados almacenados en el ArrayList, para cada una de las entradas dadas por el usuario.
4.

```
// Basado en el ejercicio en linea por Mateo Florez
public static void main (String args [])  throws IOException {

   Scanner lector=new Scanner(System.in); //c
   int n=0; //c
   int d=0; //c
   int r=0; //c
   ArrayList resp = new ArrayList<Integer>(); //c
```

```
while (((n=lector.nextInt())!=0 && (d=lector.nextInt())!=0 && (r=lector.nextInt())!=0)){ //n
   if ((n<=100)&&(d<=10000)&&(r<=5)){ //c
      BufferedReader lector2 = new BufferedReader(new InputStreamReader(System.in)); //c
      String rutaM= lector2.readLine(); //c
      String[] duracionRutaM = rutaM.split(" "); //c
      String rutaT= lector2.readLine(); //c
      String[] duracionRutaT = rutaT.split(" "); //c

      //horas totales recorridas de rutas
      int totalM=0; //c
      int totalT=0; //c
      for(int i=0; i<n; i++){ //n
         totalM+=Integer.parseInt(duracionRutaM[i]); //c
         totalT+=Integer.parseInt(duracionRutaT[i]); //c
      }

      //valor de horas extra
      int valorM=(totalM-d)*r; //c
      int valorT=(totalT-d)*r; //c
      int elmeroTotal= valorM + valorT; //c

      resp.add(elmeroTotal); //O(1)
   }else{
      System.out.println("algunos datos no cumplen las condiciones dadas"); //c
   }

}

for (int i=0; i<resp.size();i++){ //m
   System.out.println(resp.get(i)); //c
}
```

T(n)= c+n*(n+c')+c''*m
T(n)= n*(n+c')+c''*m R.S.
T(n)= n*(n+c') R.S.
T(n)= n^2 + n*c'
T(n)= n^2 R.S.
O(n)= n^2

5. La variable n hace referencia a la cantidad de rutas que se realizarán ya sea en la mañana o en la tarde, cuya duración será sumada para poder establecer el costo de las horas extras, la variable m (cuya intervención no llega al cálculo final porque es menor que los demás cálculos) hace referencia a la cantidad de respuestas, según entradas dadas por el usuario.

4) Simulacro de Parcial

1. *i=j;*
2. min > adjacencyMatrix[element][i]

3. ***a)***

Paso	a	B	C	D	E	F	G	H
1	A	20 - A	∞	80 – A	∞	∞	90 - A	∞
2	B	20 - A	∞	80 – A	∞	30 - B	90 - A	∞
3	F	20 - A	40 – F	70 – F	∞	30 - B	90 - A	∞
4	C	20 - A	40 – F	50 – C	∞	30 - B	90 - A	60 – C
5	D	20 - A	40 – F	50 – C	∞	30 - B	70 - D	60 – C
6	H	20 - A	40 – F	50 – C	∞	30 - B	70 - D	60 – C
7	G	20 - A	40 – F	50 – C	∞	30 - B	70 - D	60 – C
8	E	20 - A	40 – F	50 – C	∞	30 - B	70 - D	60 – C

b) A, B, F, C, D, G cuyo costo es 70

DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co